ALCF COMPUTATIONAL PERFORMANCE WORKSHOP

VISUALIZATION INTRODUCTION

JOE INSLEY
Lead, Visualization & Data Analytics
Argonne Leadership Computing Facility

SILVIO RIZZI
Assistant Computer Scientist
Argonne Leadership Computing Facility

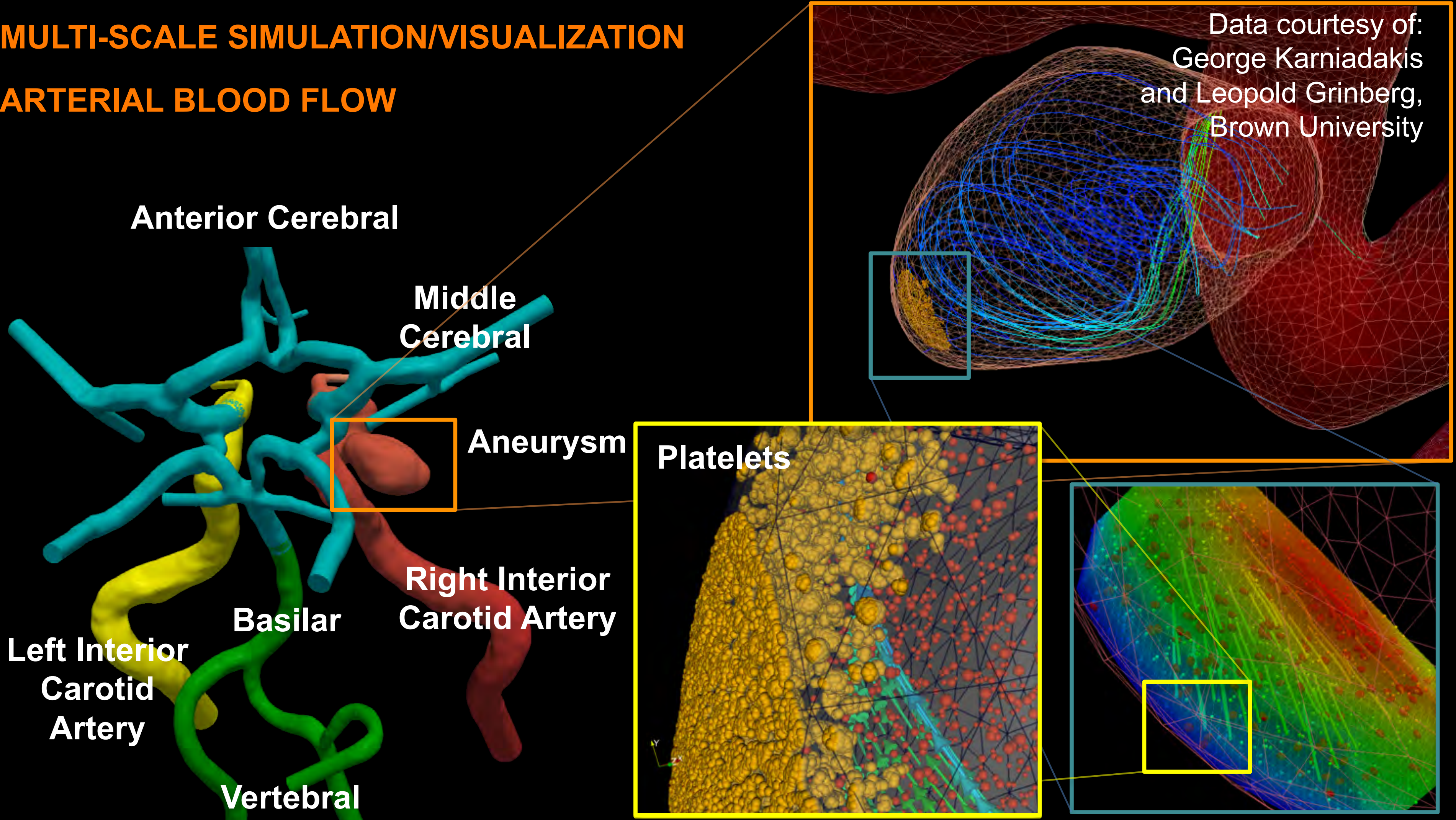
May 4th, 2017
Argonne National Laboratory

HERE'S THE PLAN…

- Examples of visualizations
- Visualization resources
- Visualization tools and formats
- Data representations
- Annotation and movie creation
- Visualization for debugging
- In-Situ Visualization and Analysis

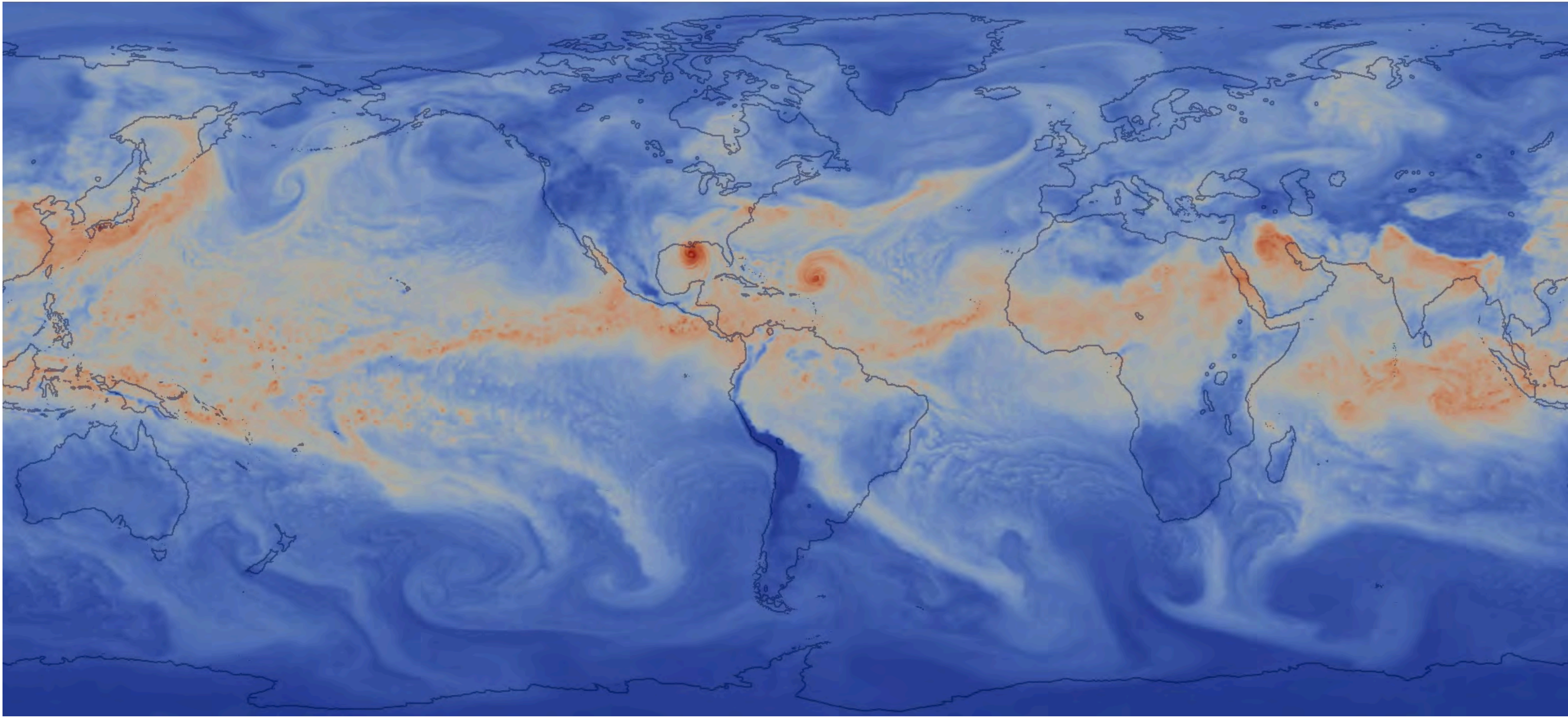
MULTI-SCALE SIMULATION/VISUALIZATION

ARTERIAL BLOOD FLOW



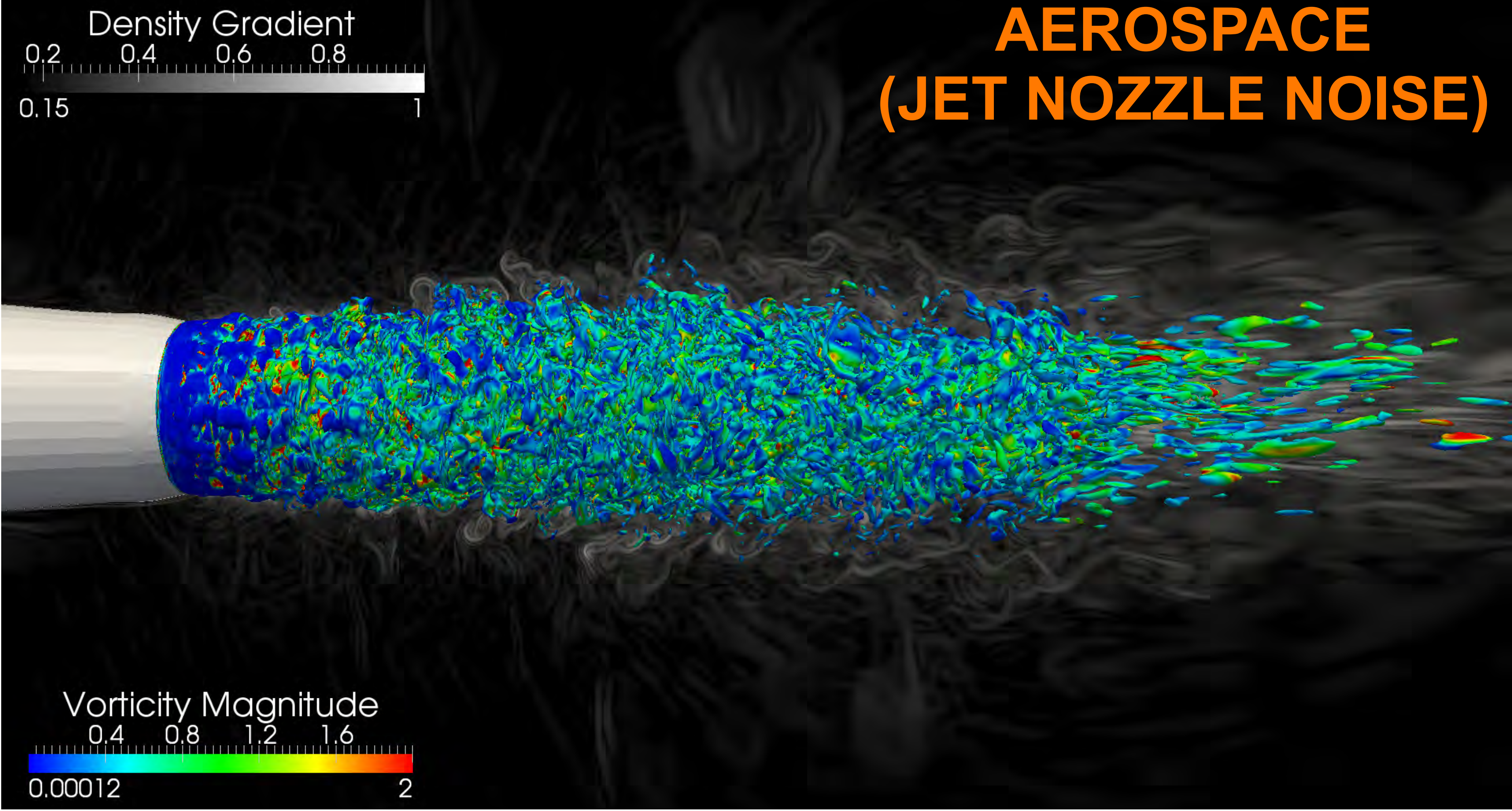
Data courtesy of:
George Karniadakis
and Leopold Grinberg,
Brown University

CLIMATE



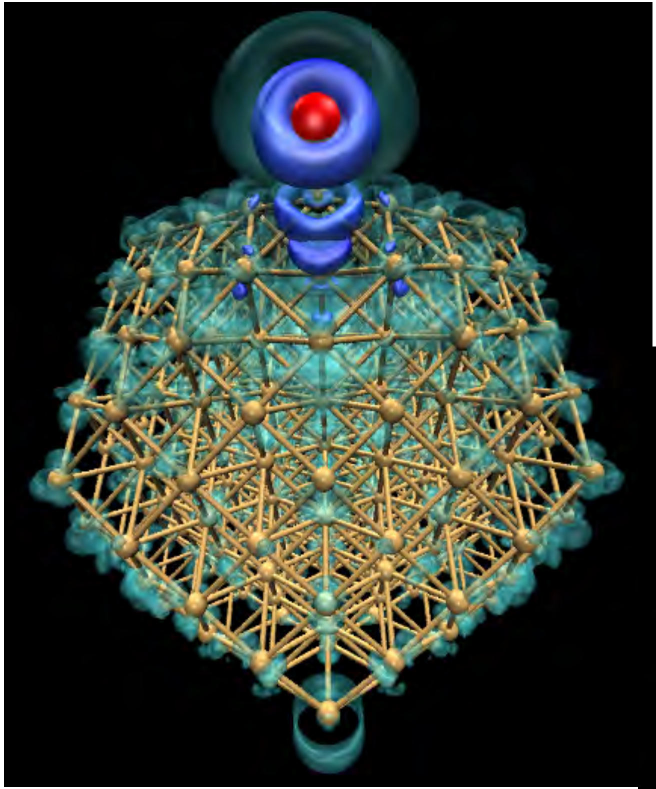
Data courtesy of: Mark Taylor, Sandia National Laboratory; Rob Jacob, Argonne National Laboratory; Warren Washington, National Center for Atmospheric Research

AEROSPACE (JET NOZZLE NOISE)

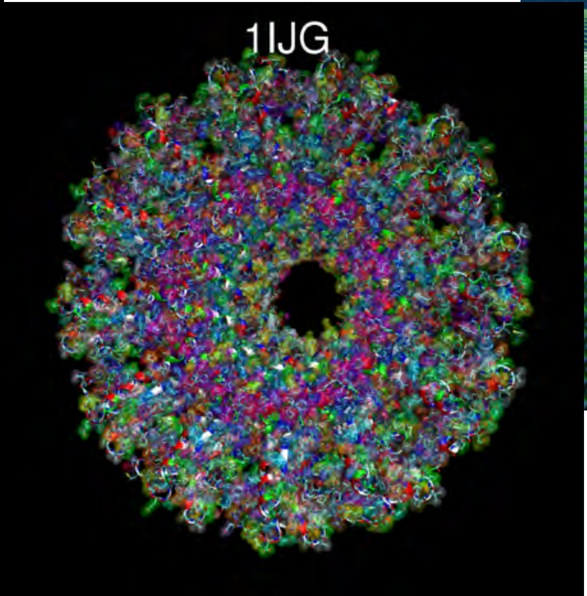


Data courtesy of: Anurag Gupta and Umesh Paliath, General Electric Global Research

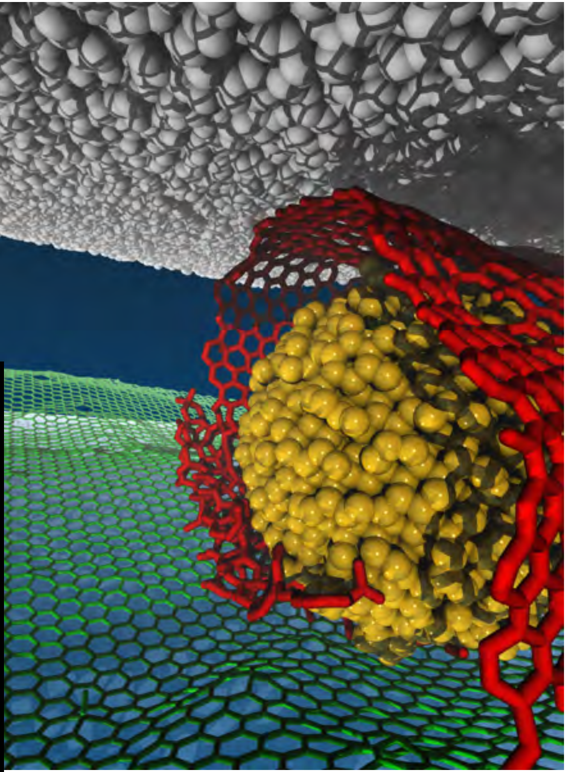
MATERIALS SCIENCE / MOLECULAR



Data courtesy of: Jeff Greeley, Nichols Romero, Argonne National Laboratory

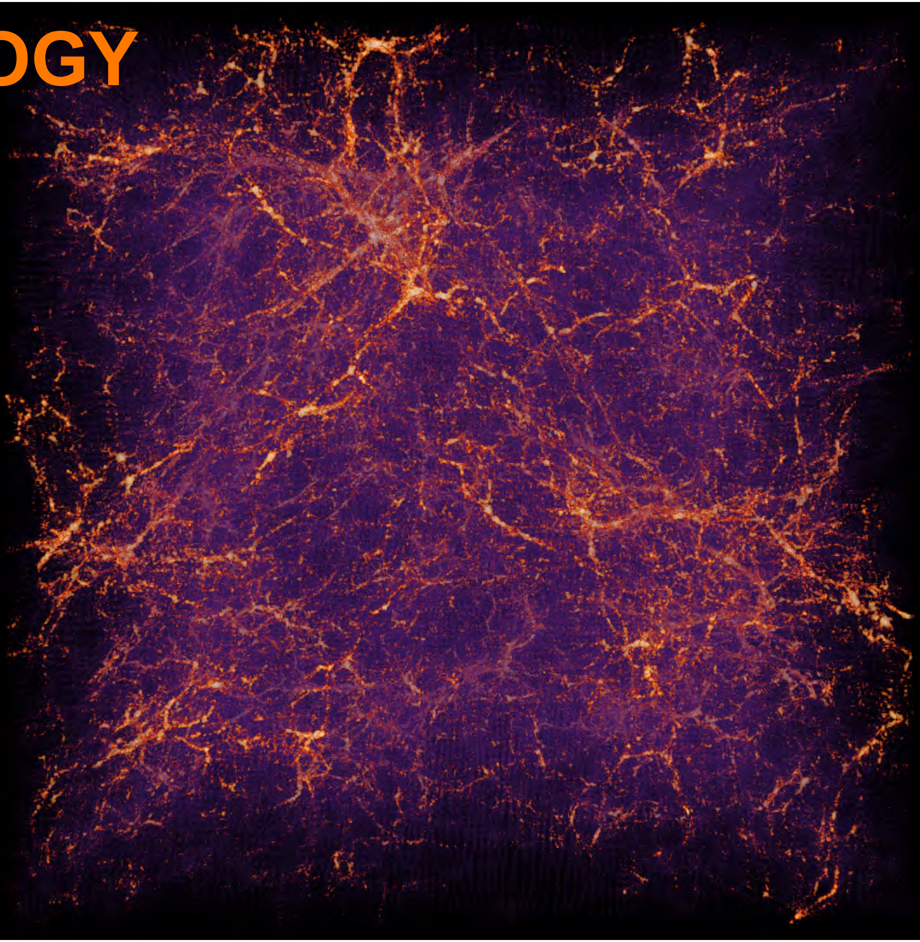


Data courtesy of: Subramanian Sankaranarayanan, Argonne National Laboratory



Data courtesy of: Advanced Photon Source, Argonne National Laboratory

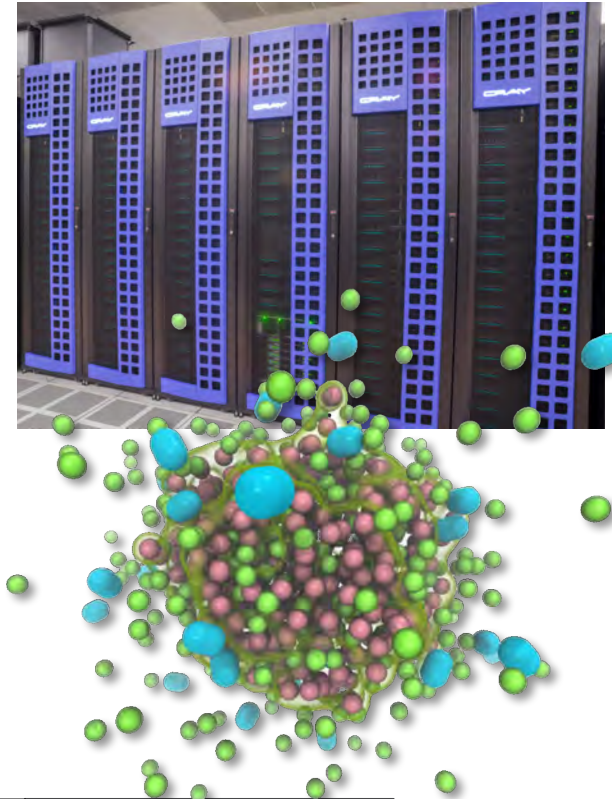
COSMOLOGY

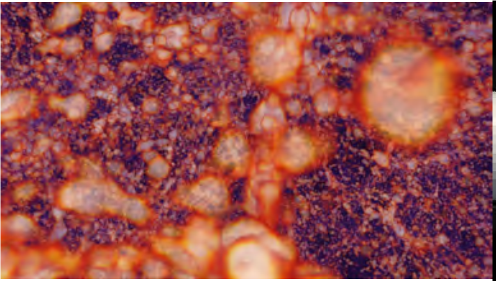


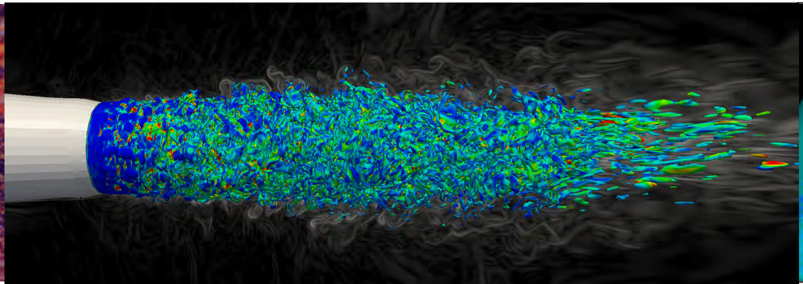
Data courtesy of: Salman Habib, Katrin Heitmann, and the HACC team, Argonne National Laboratory

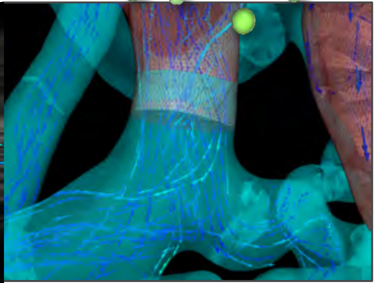
COOLEY

- Analytics/Visualization cluster
- Peak 223 TF
- 126 nodes; each node has
  - Two Intel Xeon E5-2620 Haswell 2.4 GHz 6-core processors
  - NVIDIA Telsa K80 graphics processing unit (24GB)
  - **384 GB of RAM**
- Aggregate RAM of 47 TB (vs. ~6TB for Tukey)
- Aggregate GPU memory of~3TB (vs. ~1.1TB for Tukey)
- Cray CS System
- 216 port FDR IB switch with uplinks to our QDR infrastructure
- Mounts the same GPFS file systems as Mira, Cetus









VISUALIZATION TOOLS AND DATA FORMATS

ALL SORTS OF TOOLS

- Visualization Applications
  - VisIt
  - ParaView
  - EnSight
- Domain Specific
  - VMD, PyMol, RasMol
- APIs
  - VTK: visualization
  - ITK: segmentation & registration
- GPU performance
  - vl3: shader-based volume rendering
- Analysis Environments
  - Matlab
  - Parallel R
- Utilities
  - GnuPlot
  - ImageMagick
- Visualization Workflow
  - VisTrails

PARAVIEW & VISIT VS. VTK

- ParaView & VisIt
  - General purpose visualization applications
  - GUI-based
  - Scriptable
  - Extendable
  - Built on top of vtk (largely)

- vtk
  - Programming environment / API
  - Additional capabilities, finer control
  - Smaller memory footprint
  - Requires more expertise (build custom applications)

DATA FILE FORMATS (PARAVIEW & VISIT)

- VTK
- Parallel (partitioned) VTK
- VTK MultiBlock (MultiGroup, Hierarchical, Hierarchical Box)
- Legacy VTK
- Parallel (partitioned) legacy VTK
- EnSight files
- EnSight Master Server
- Exodus
- BYU
- XDMF
- PLOT2D
- PLOT3D
- SpyPlot CTH
- HDF5 raw image data
- DEM
- VRML
- PLY
- Polygonal Protein Data Bank
- XMol Molecule
- Stereo Lithography
- Gaussian Cube
- Raw (binary)
- AVS
- Meta Image
- Facet
- PNG
- SAF
- LS-Dyna
- Nek5000
- OVERFLOW
- paraDIS
- PATRAN
- PFLOTRAN
- Pixie
- PuReMD
- S3D
- SAS
- Tetrad
- UNIC
- VASP
- ZeusMP
- ANALYZE
- BOV
- GMV
- Tecplot
- Vis5D
- Xmdv
- XSF

DATA WRANGLING

- XDMF
  - XML wrapper around HDF5 data
  - Can define
    - data sets
    - subsets
    - hyperslabs
- vtk
  - Could add to your simulation code
  - Can write small utilities to convert data
    - Use your own read routines
    - Write vtk data structures
  - C++ and Python bindings

DATA ORGANIZATION

- Format
  - Existing tools support many flavors
  - Use one of these formats
  - Use (or write) a format converter
  - Write a custom reader for existing tool
  - Write your own custom vis tool

- Serial vs. Parallel/Partitioned
  - Single big file vs. many small files: middle ground generally best
    - vtk data types
    - XDMF for HDF5 (VisIt and ParaView)
    - Custom

DATA ORGANIZATION

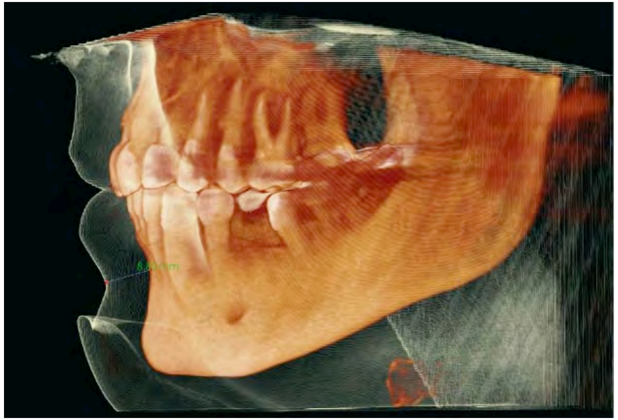
- Serial vs. Parallel/Partitioned
  - Performance trade-offs
    - vtk/paraview: serial files all data read on head node, partitioned and distributed
    - vtk/paraview: parallel files: serial files partitioned

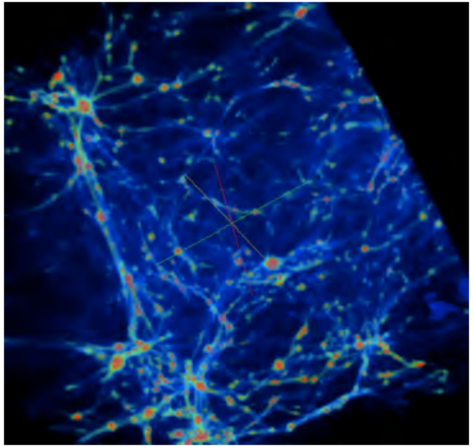
Performance example:

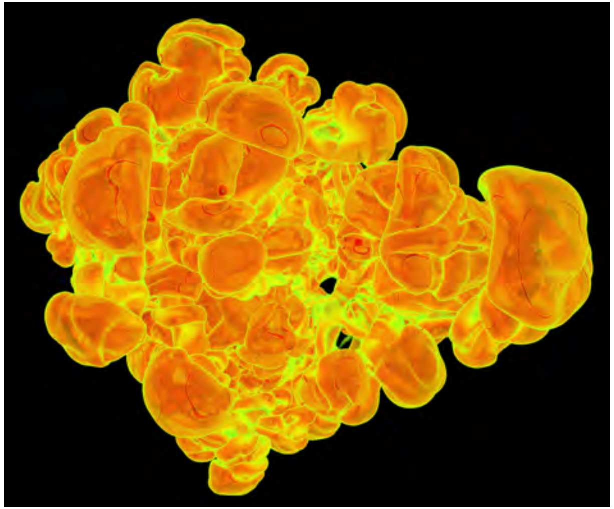
- Single serial .vtu file (unstructured grid)
  - Data size: ~3.8GB
  - Read time on 64 processes: > 15 minutes
    - most of this was spent partitioning and distributing
- Partitioned .pvtu file (unstructured grid)
  - Data size: ~8.7GB (64 partitions)
  - Read time on 64 processes: < 1 second

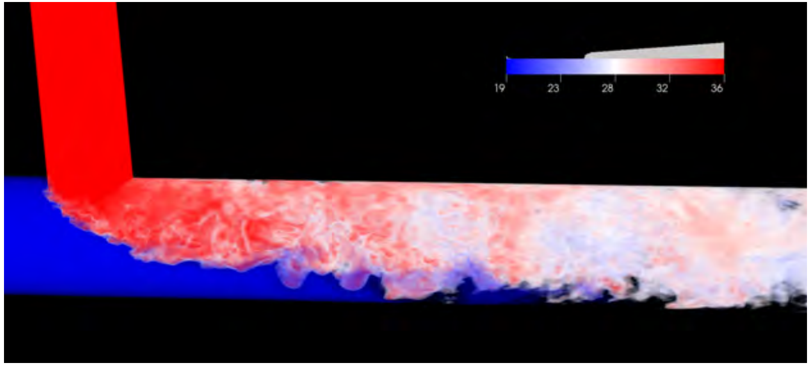
DATA REPRESENTATIONS

DATA REPRESENTATIONS: VOLUME RENDERING



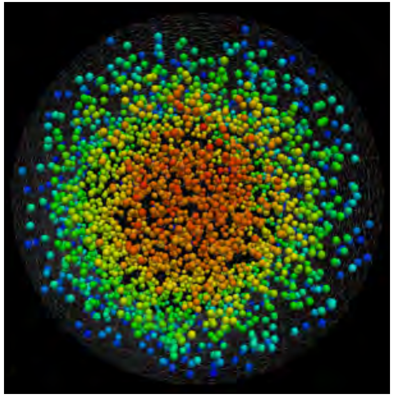


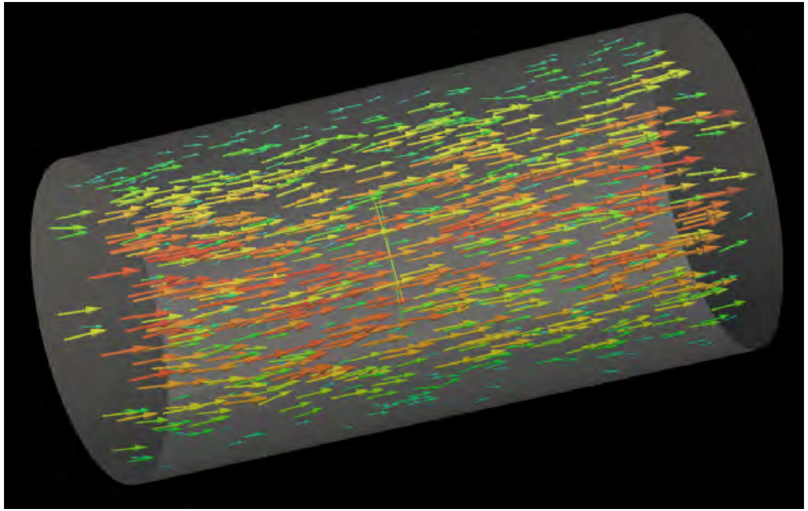


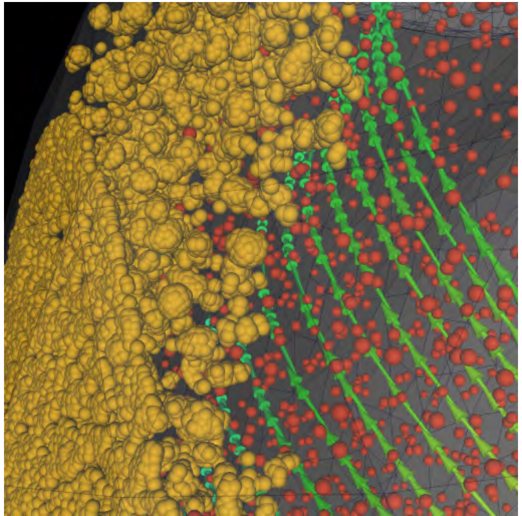


DATA REPRESENTATIONS: GLYPHS

- 2D or 3D geometric object to represent point data
- Location dictated by coordinate
  - 3D location on mesh
  - 2D position in table/graph
- Attributes graphical entity dictated by attributes of a data
  - color, size, orientation

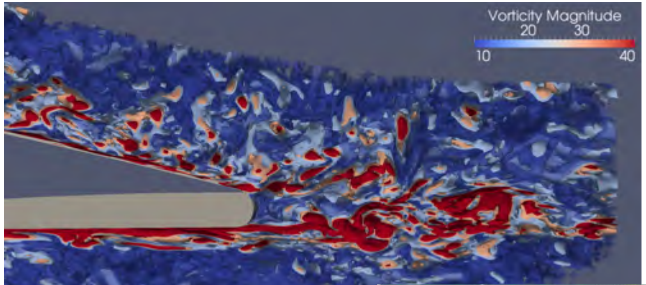


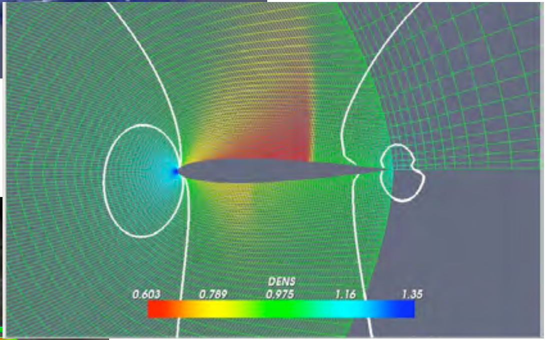


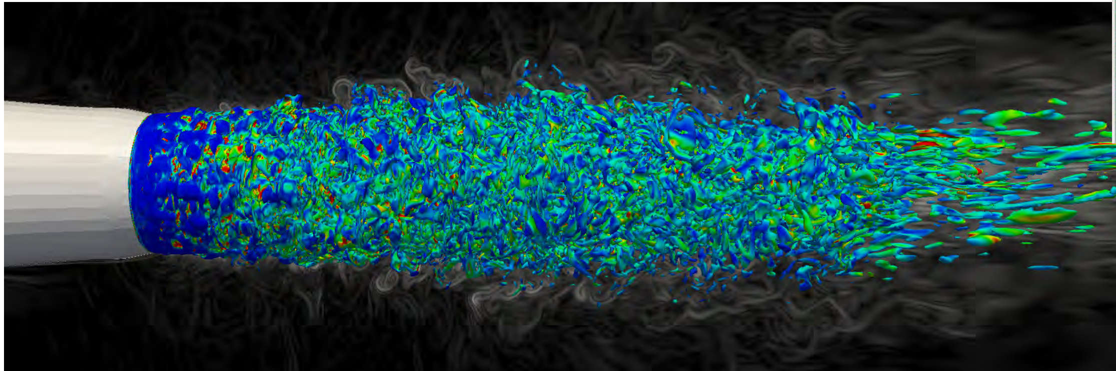


DATA REPRESENTATIONS: CONTOURS (ISOSURFACES)

- A Line (2D) or Surface (3D), representing a constant value
- VisIt & ParaView:
  - good at this
- vtk:
  - same, but again requires more effort

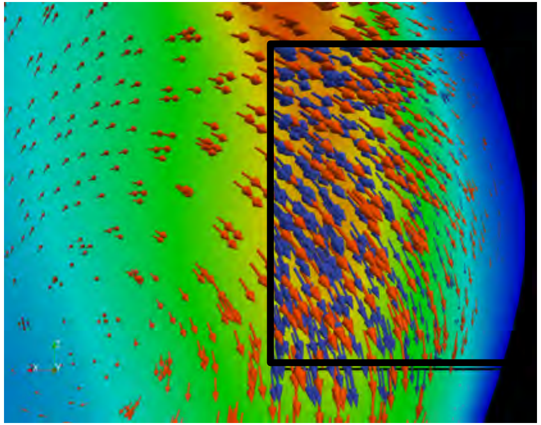


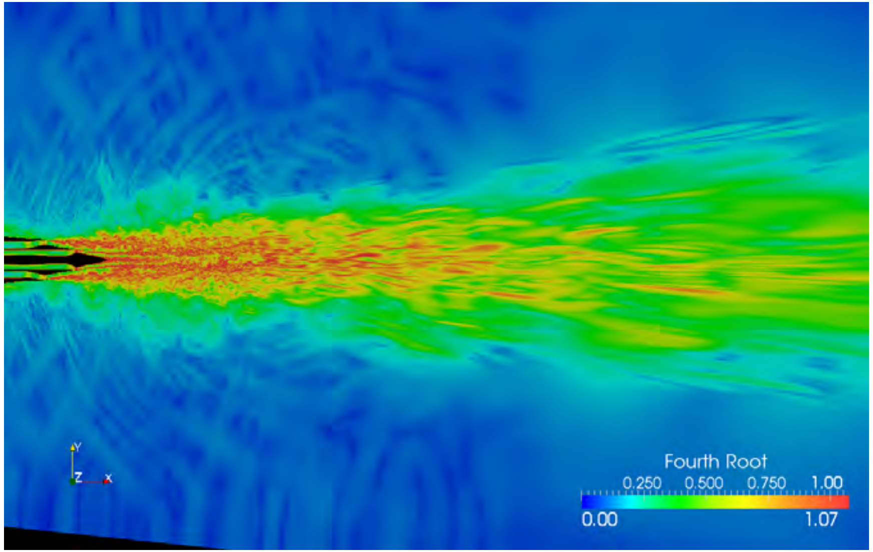




DATA REPRESENTATIONS: CUTTING PLANES

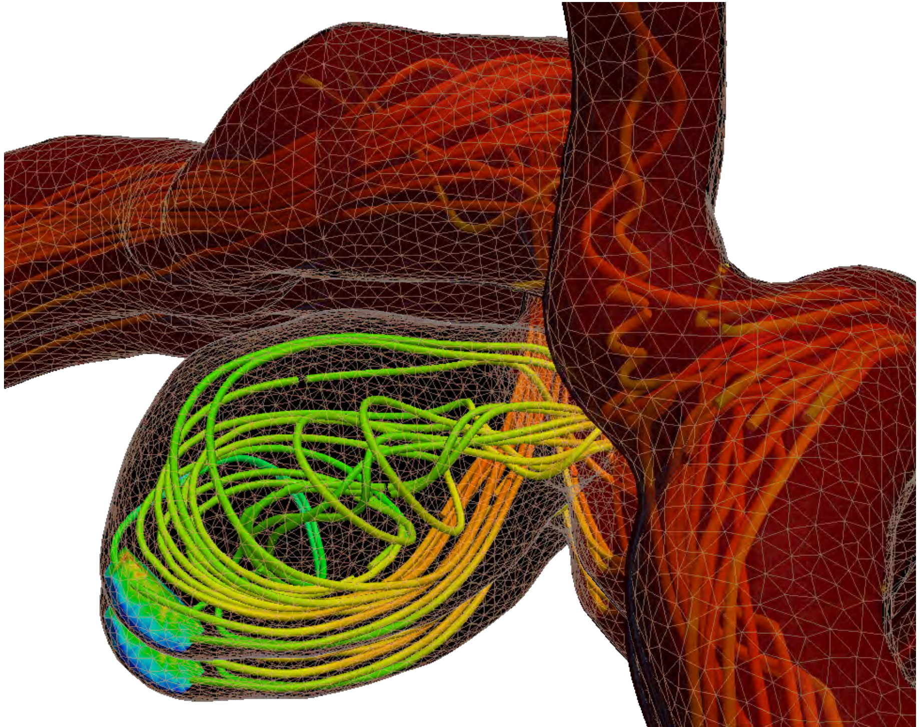
- Slice a plane through the data
  - Can apply additional visualization methods to resulting plane
- VisIt & ParaView & vtk good at this
- VMD has similar capabilities for some data formats

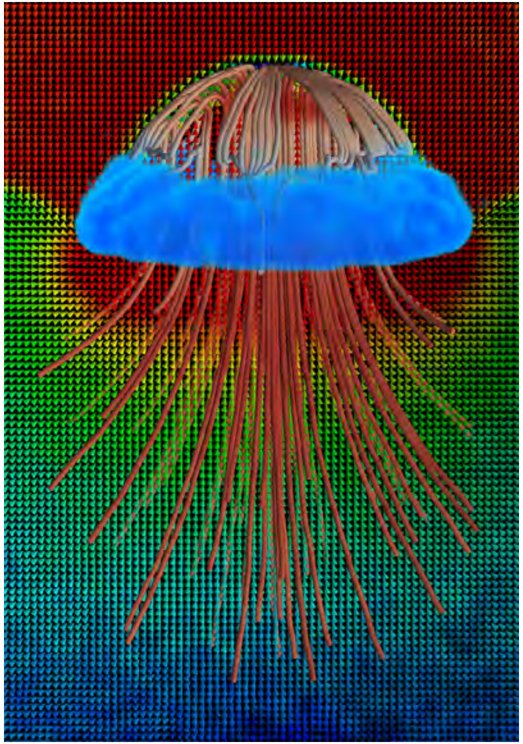




DATA REPRESENTATIONS: STREAMLINES

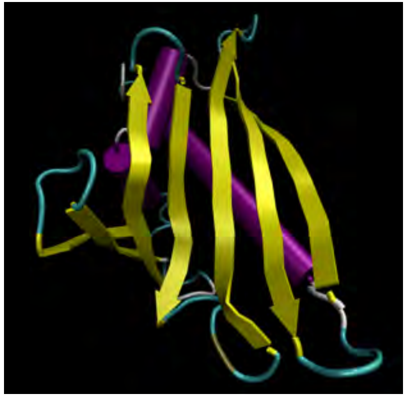
- From vector field on a mesh (needs connectivity)
  - Show the direction an element will travel in at any point in time.
- VisIt & ParaView & vtk good at this

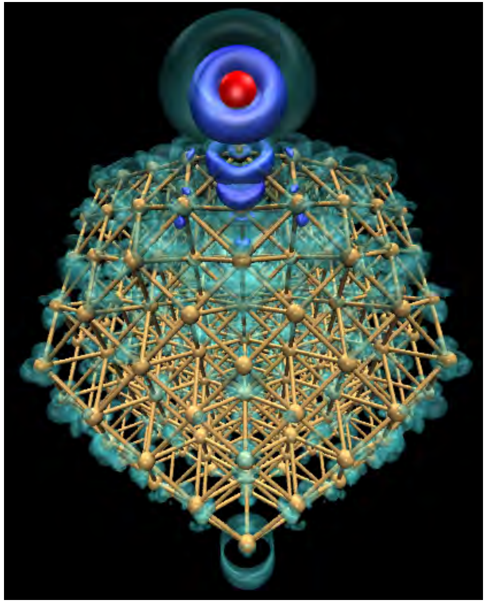


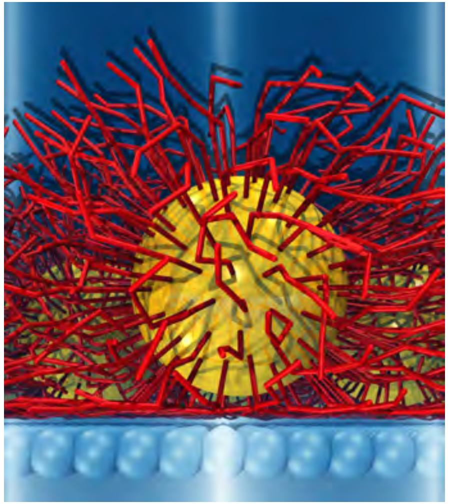


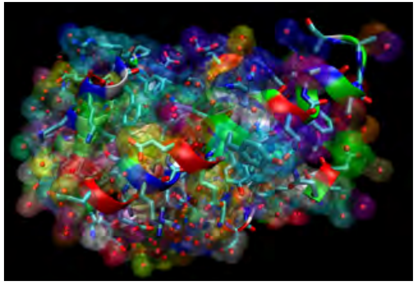
MOLECULAR DYNAMICS VISUALIZATION

- VMD:
  - Lots of domain-specific representations
  - Many different file formats
  - Animation
  - Scriptable
  - Not parallel
- VisIt & ParaView:
  - Limited support for these types of representations
- VTK:
  - Anything’s possible if you try hard enough





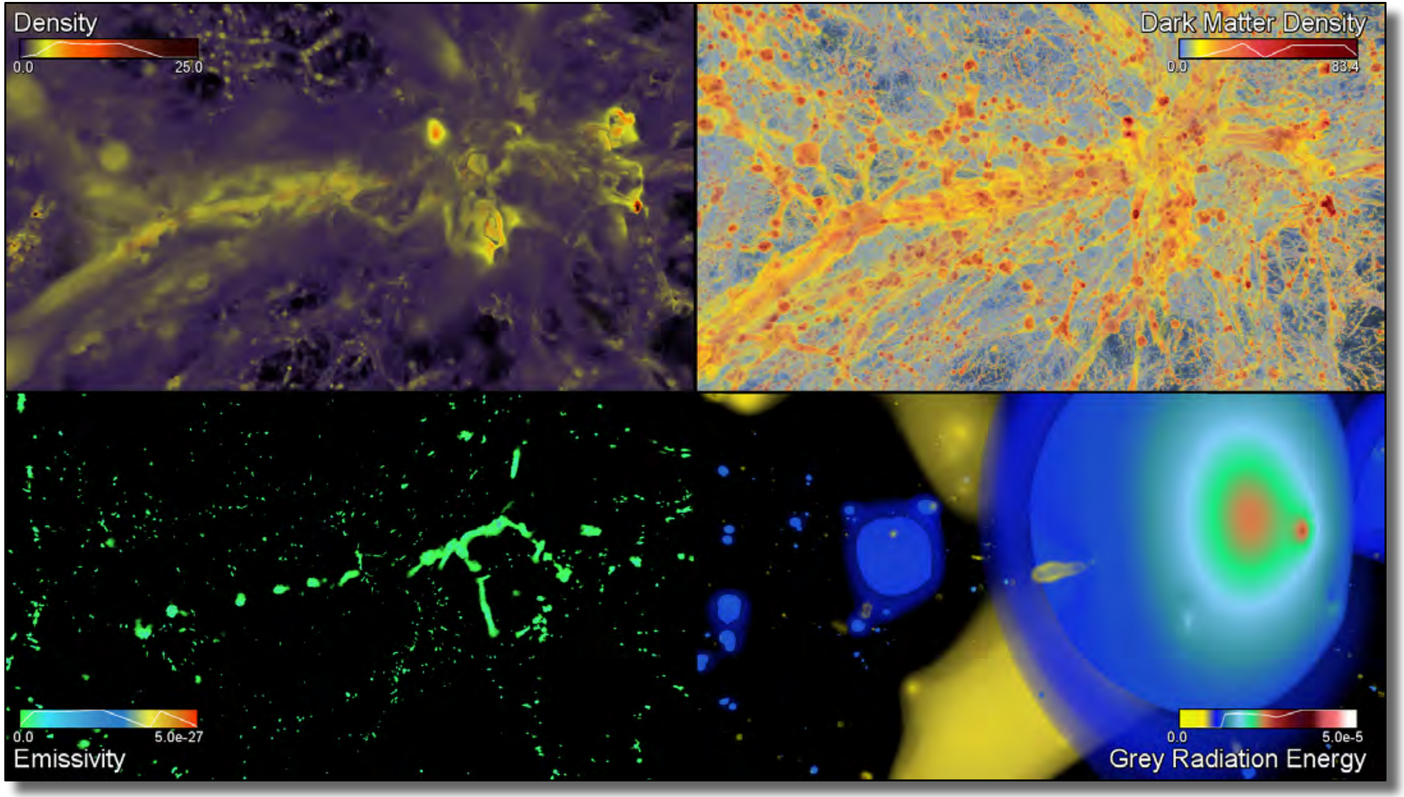




ANNOTATION AND MOVIE CREATION

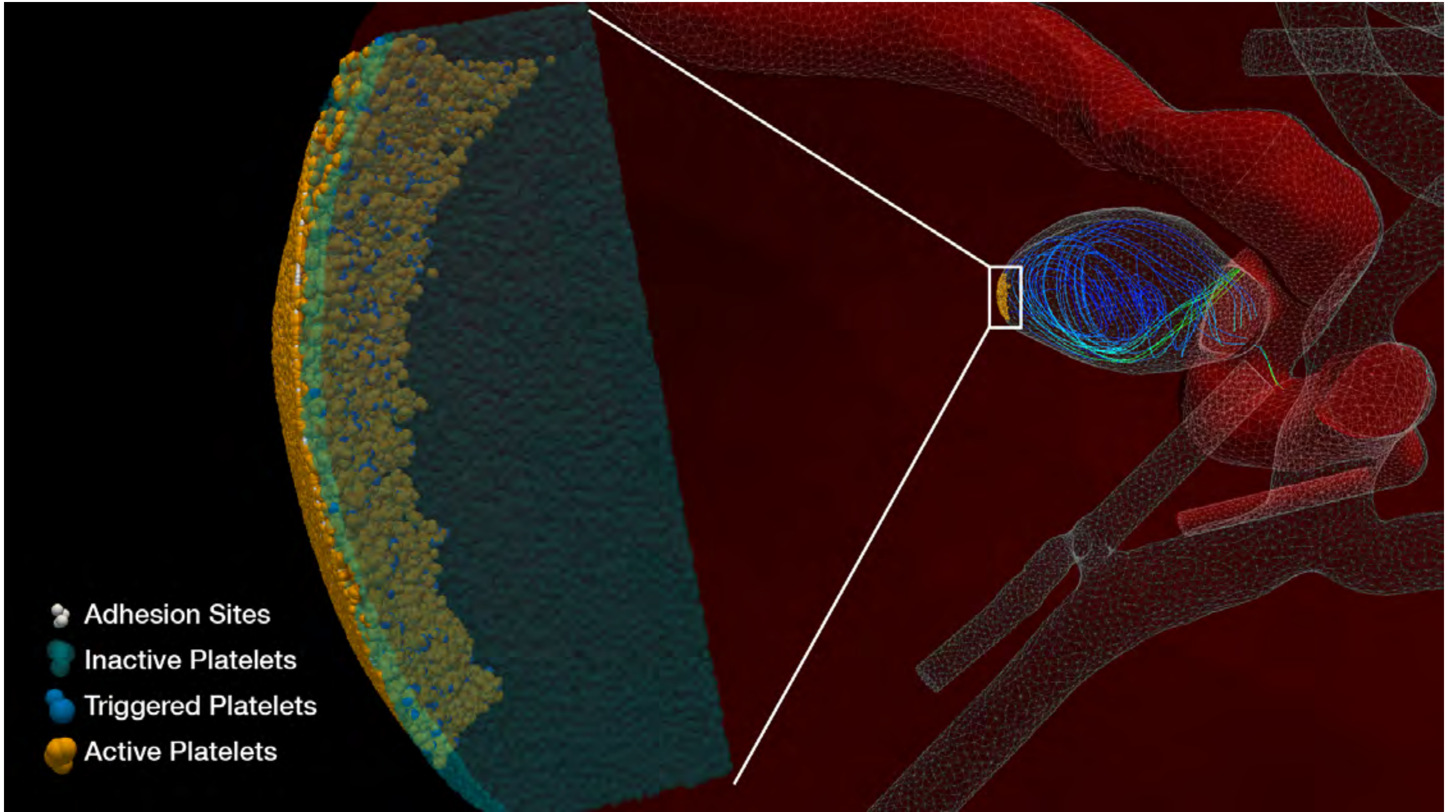
ANNOTATION, COMPOSITING, SCALING…

- ImageMagick
  - convert, composite, montage, etc.



ANNOTATION, COMPOSITING, SCALING…

- ImageMagick
  - scale, fade

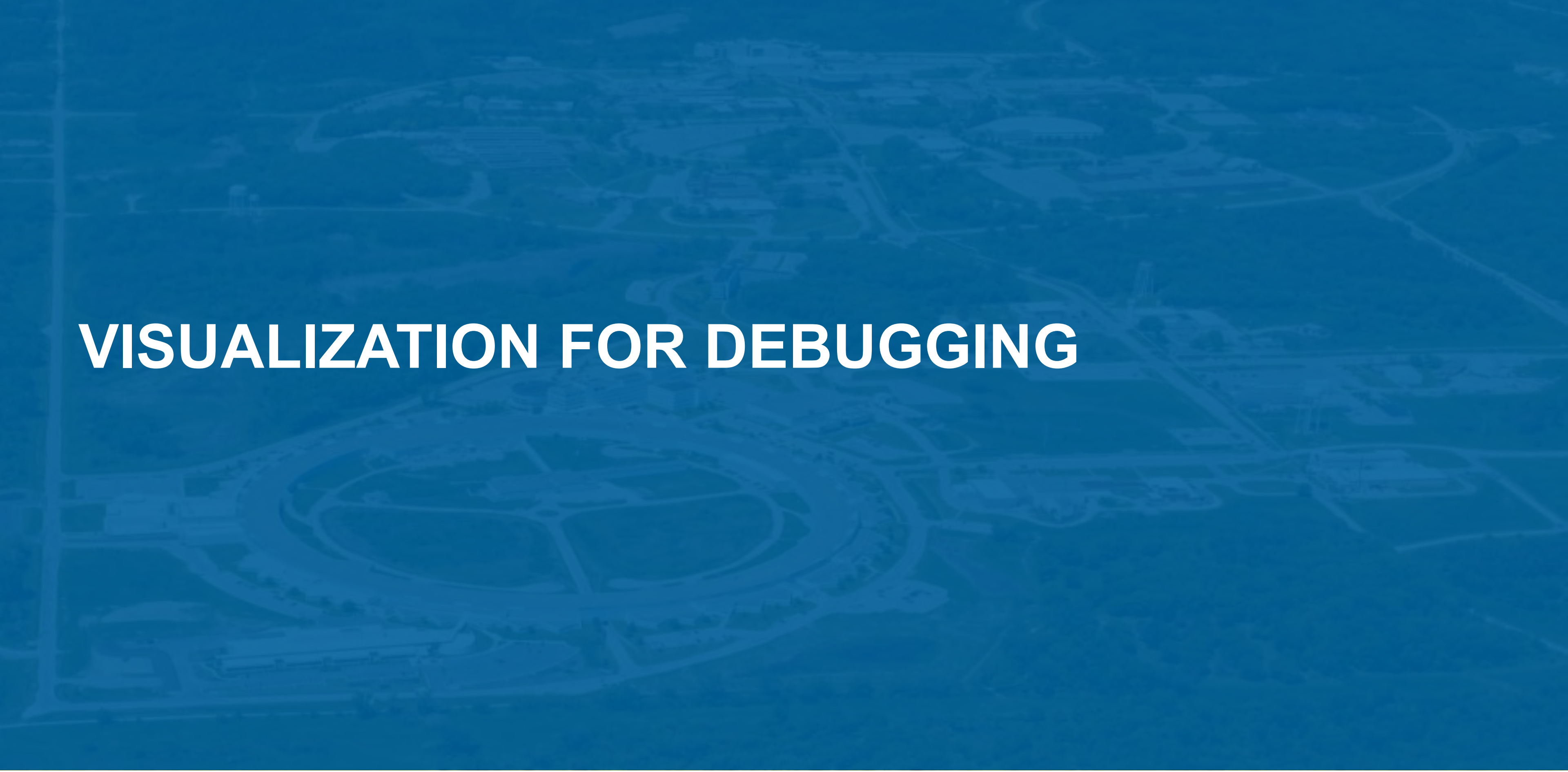


MOVIE CREATION

- VisIt and ParaView can spit out a movie file (.avi, etc.)
  - can also spit out individual images

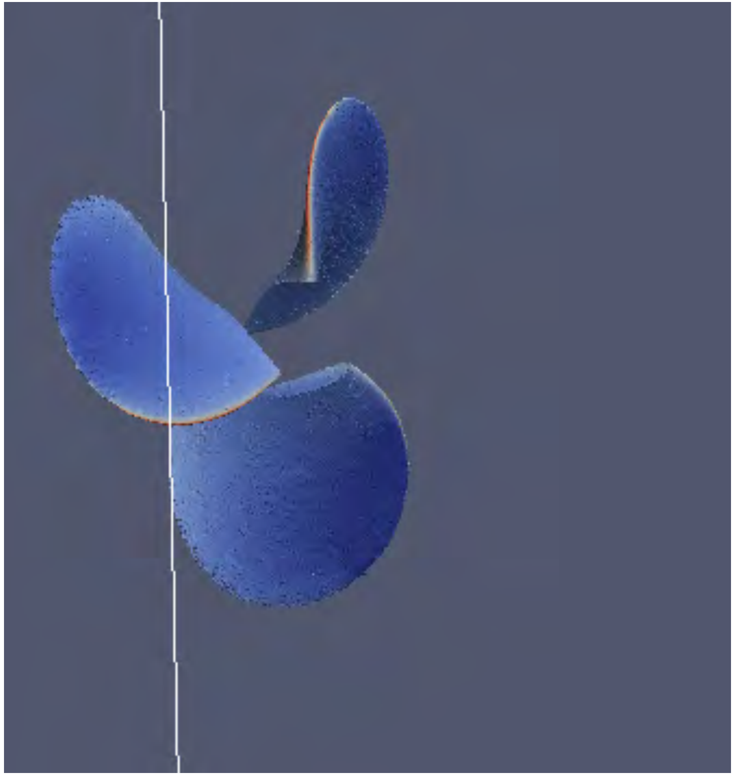
- Combine multiple segments of frames
  - Create a directory of symbolic links to all frames in order

- ffmpeg: Movie encoding
  - ffmpeg –sameq –i frame.%04d.png movie.mp4

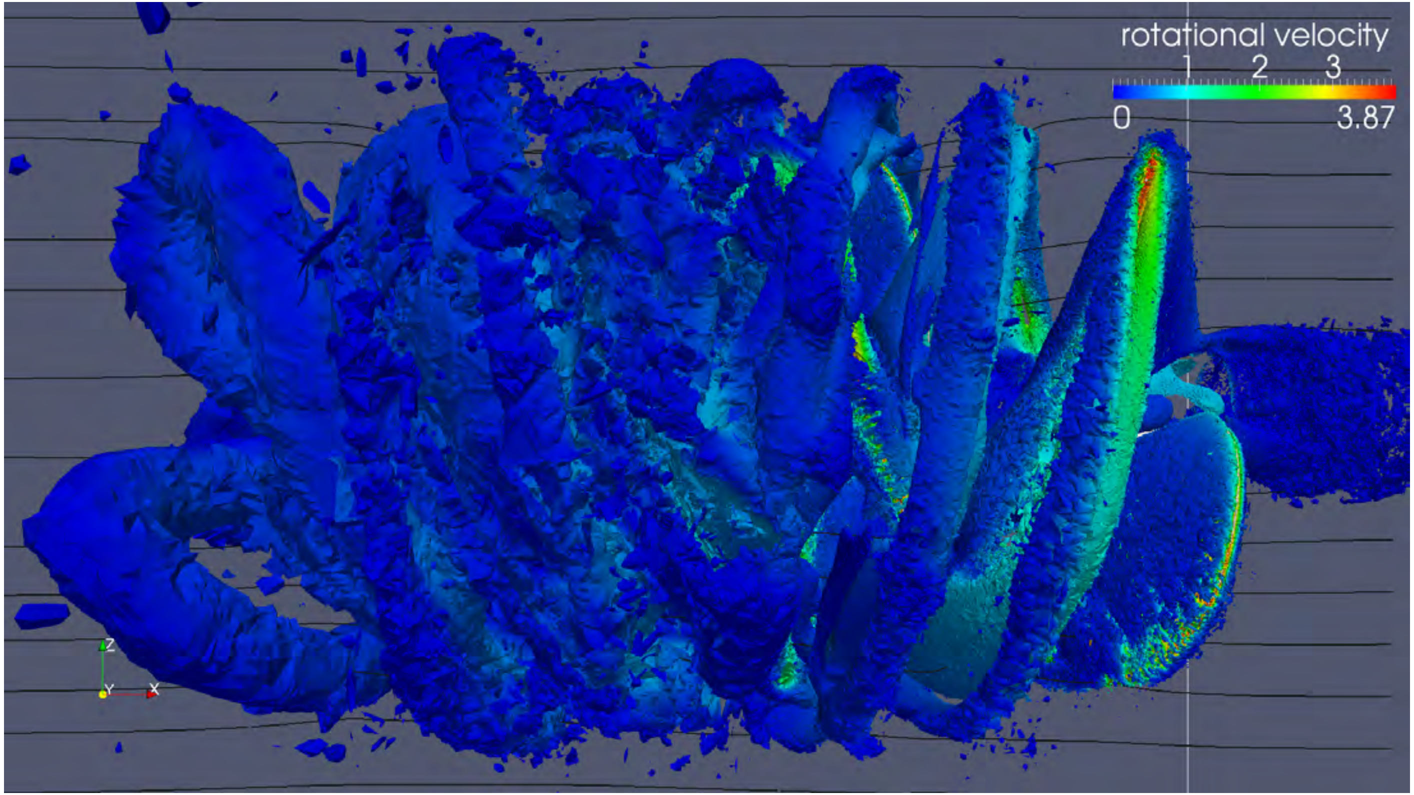


VISUALIZATION FOR DEBUGGING

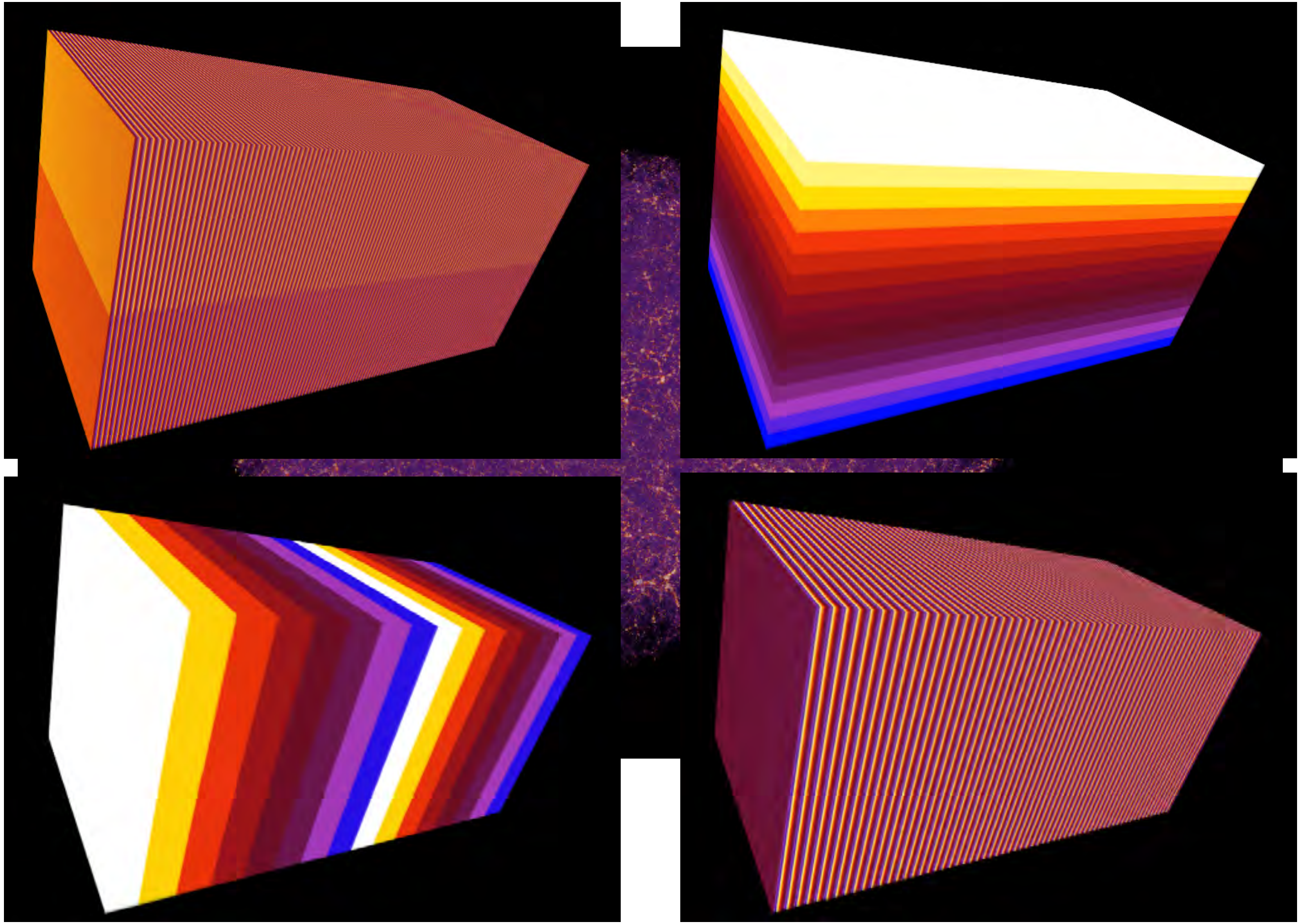
VISUALIZATION FOR DEBUGGING



VISUALIZATION FOR DEBUGGING

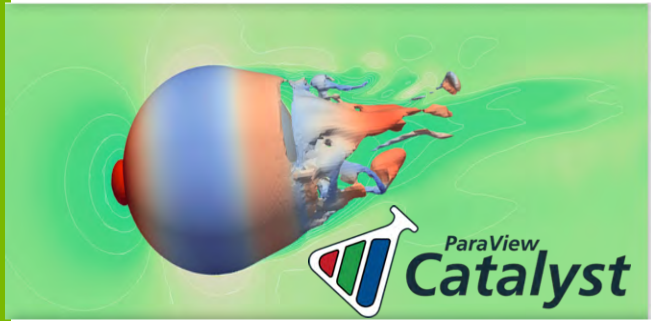


VISUALIZATION AS DIAGNOSTICS: COLOR BY THREAD ID



IN-SITU VISUALIZATION AND ANALYSIS

MULTIPLE IN-SITU INFRASTRUCTURES



LibSim

CAN WE….

- Enable use of any in situ framework?
- Develop analysis routines that are portable between codes?
- Make it easy to use?

OUR APPROACH

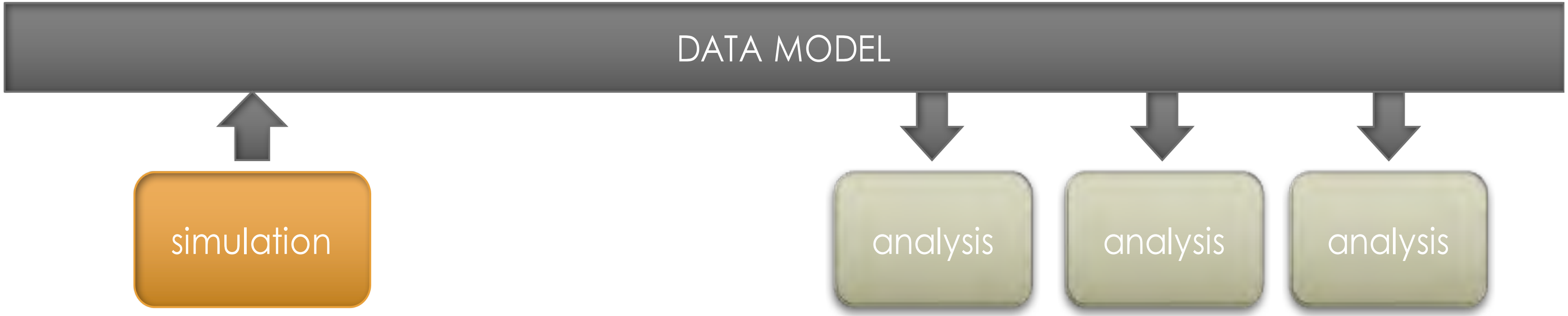
- Data model – to pass data between Simulation & Analysis
- API – for instrumenting simulation and analysis codes

SENSEI
insitu

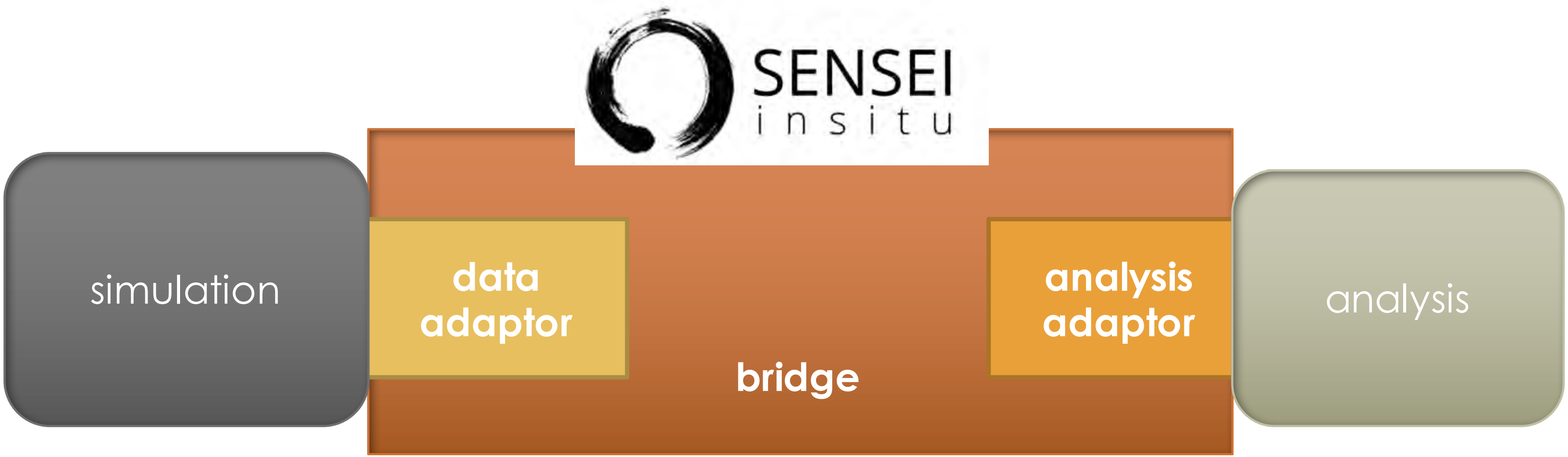
DATA MODEL: VTK

- Used by ParaView/Catalyst and VisIt/Libsim
- Supports common scientific dataset types
- On going independent efforts to evolve for exascale
- Supports using simulation memory directly (zero-copy) for multiple memory layouts

http://www.vtk.org/



SENSEI: API: COMPONENTS



INSTRUMENTATION TASKS

FOR SIMULATION

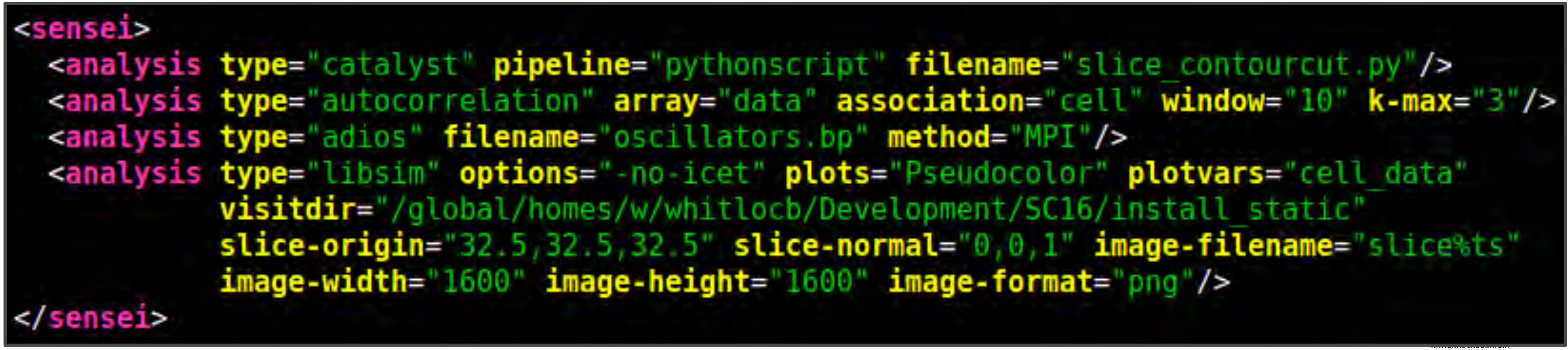
- Write a Data Adaptor to map simulation data to VTK data model
- Write a Bridge to define API entry points for simulation

FOR ANALYSIS

- Write analysis adaptor that uses Data Adaptor API to access Data
- Transform data, if needed and invoke analysis

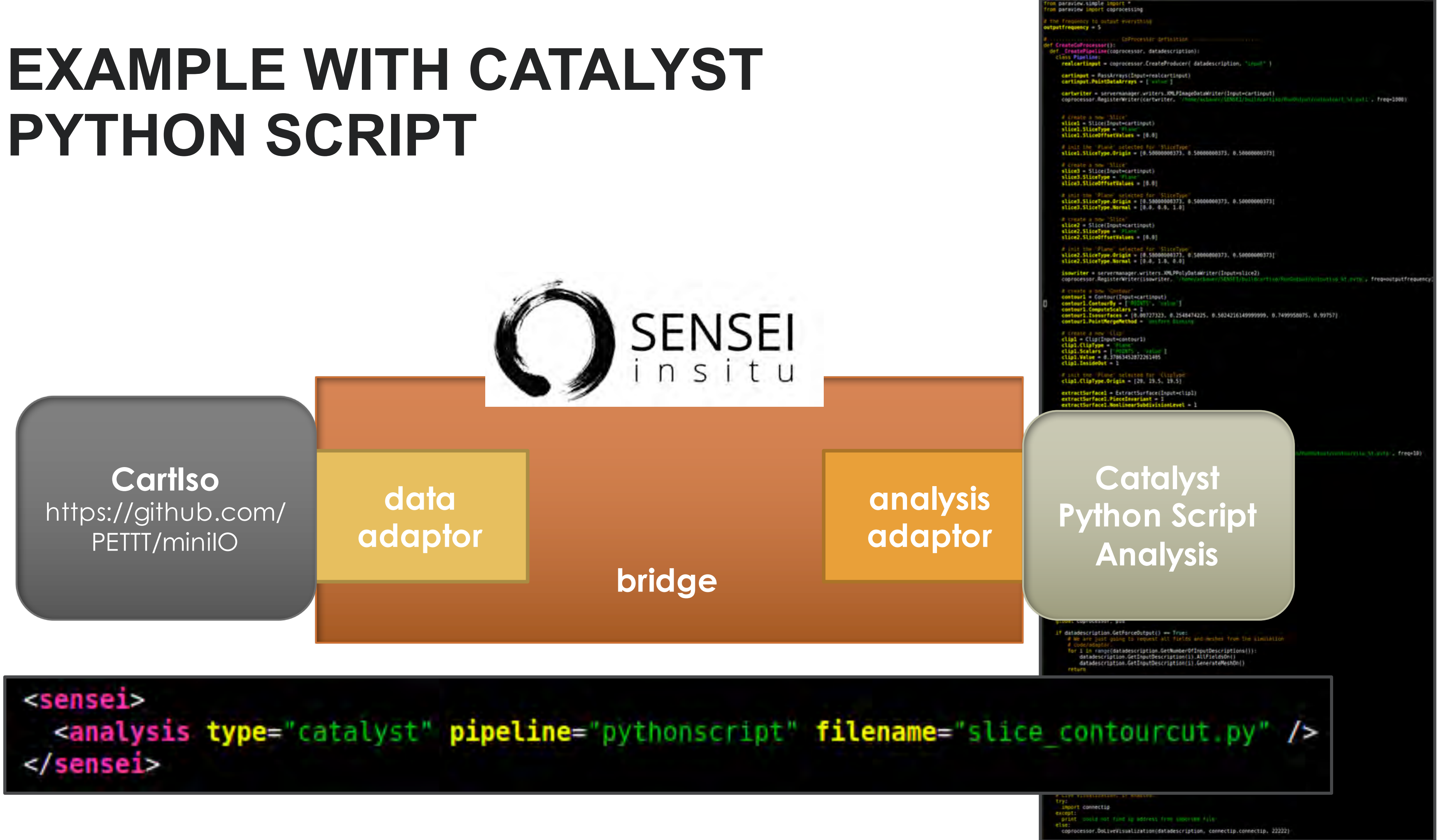
ADDING A CATALYST PYTHON SCRIPT ANALYSIS

- 13 lines of CMake code changes
- 18 lines of C++ code
- In situ work can be specified via SENSEI XML



```
<sensei>
  <analysis type="catalyst" pipeline="pythonscript" filename="slice_contourcut.py"/>
  <analysis type="autocorrelation" array="data" association="cell" window="10" k-max="3"/>
  <analysis type="adios" filename="oscillators.bp" method="MPI"/>
  <analysis type="libsim" options="-no-icet" plots="Pseudocolor" plotvars="cell_data"
            visitdir="/global/homes/w/whitlocb/Development/SC16/install_static"
            slice-origin="32.5,32.5,32.5" slice-normal="0,0,1" image-filename="slice%ts"
            image-width="1600" image-height="1600" image-format="png"/>
</sensei>
```

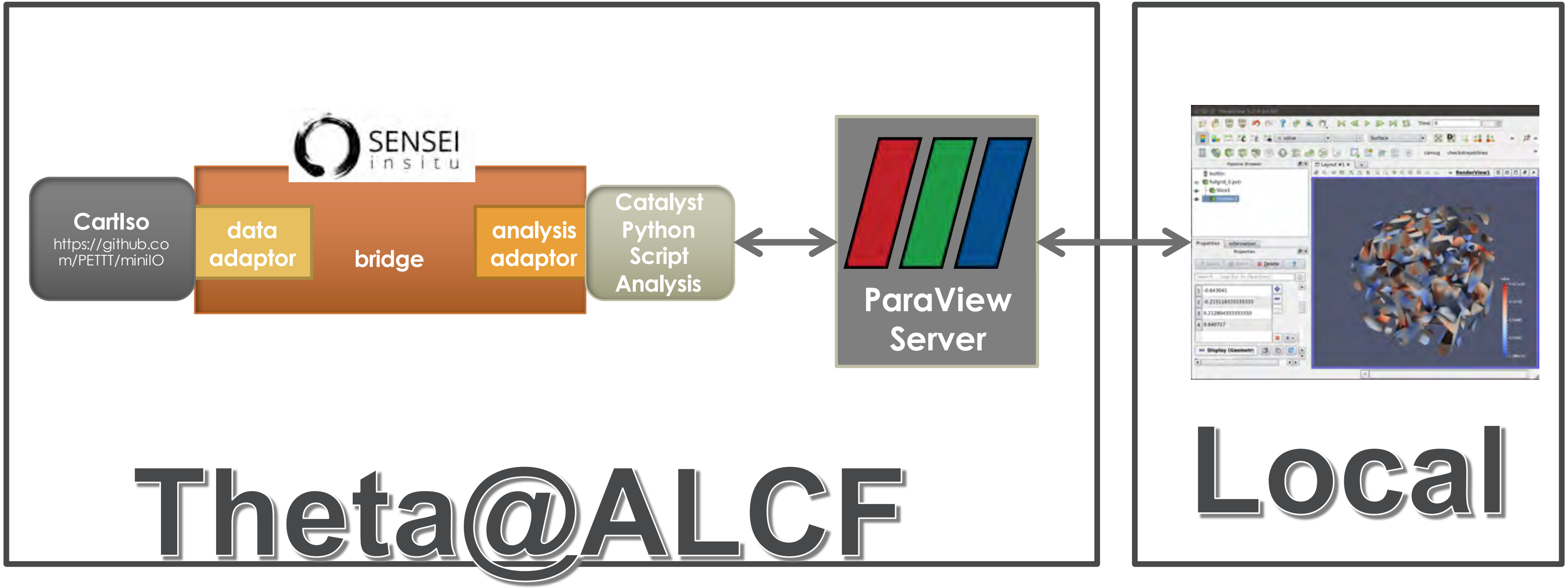
EXAMPLE WITH CATALYST PYTHON SCRIPT

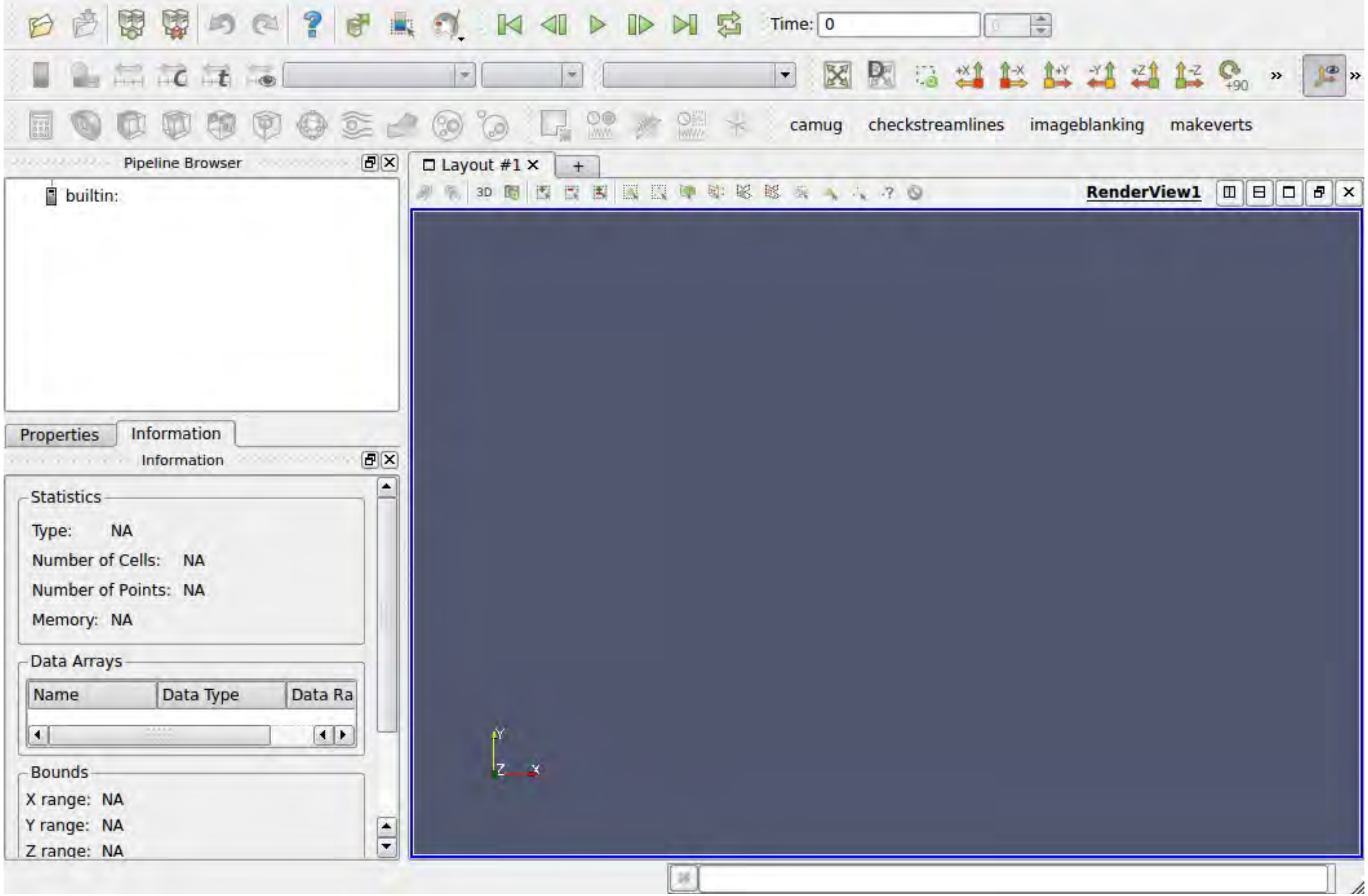

SENSEI
insitu
CartIso
https://github.com/PETTT/miniIO
data adaptor
bridge
analysis adaptor
Catalyst Python Script Analysis

```python
from paraview.simple import *
from paraview import coprocessing

# the frequency to output everything
outputfrequency = 5

# ----------------------- CoProcessor definition -----------------------
def CreateCoProcessor():
  def _CreatePipeline(coprocessor, datadescription):
    class Pipeline:
      realcartinput = coprocessor.CreateProducer( datadescription, 'input' )

      cartinput = PassArrays(Input=realcartinput)
      cartinput.PointDataArrays = [ 'value' ]

      cartwriter = servermanager.writers.XMLPImageDataWriter(Input=cartinput)
      coprocessor.RegisterWriter(cartwriter, '/home/acbauer/SENSEI/buildcartiso/RunOutput/cellsoutcart_%t.pvti', freq=1000)


      # create a new 'Slice'
      slice1 = Slice(Input=cartinput)
      slice1.SliceType = 'Plane'
      slice1.SliceOffsetValues = [0.0]

      # init the 'Plane' selected for 'SliceType'
      slice1.SliceType.Origin = [0.50000000373, 0.50000000373, 0.50000000373]

      # create a new 'Slice'
      slice3 = Slice(Input=cartinput)
      slice3.SliceType = 'Plane'
      slice3.SliceOffsetValues = [0.0]

      # init the 'Plane' selected for 'SliceType'
      slice3.SliceType.Origin = [0.50000000373, 0.50000000373, 0.50000000373]
      slice3.SliceType.Normal = [0.0, 0.0, 1.0]

      # create a new 'Slice'
      slice2 = Slice(Input=cartinput)
      slice2.SliceType = 'Plane'
      slice2.SliceOffsetValues = [0.0]

      # init the 'Plane' selected for 'SliceType'
      slice2.SliceType.Origin = [0.50000000373, 0.50000000373, 0.50000000373]
      slice2.SliceType.Normal = [0.0, 1.0, 0.0]

      isowriter = servermanager.writers.XMLPPolyDataWriter(Input=slice2)
      coprocessor.RegisterWriter(isowriter, '/home/acbauer/SENSEI/buildcartiso/RunOutput/outputiso_%t.pvtp', freq=outputfrequency)

      # create a new 'Contour'
      contour1 = Contour(Input=cartinput)
      contour1.ContourBy = ['POINTS', 'value']
      contour1.ComputeScalars = 1
      contour1.Isosurfaces = [0.00727323, 0.2548474225, 0.5024216149999999, 0.7499958075, 0.99757]
      contour1.PointMergeMethod = 'Uniform Binning'

      # create a new 'Clip'
      clip1 = Clip(Input=contour1)
      clip1.ClipType = 'Plane'
      clip1.Scalars = ['POINTS', 'value']
      clip1.Value = 0.37863452872261485
      clip1.InsideOut = 1

      # init the 'Plane' selected for 'ClipType'
      clip1.ClipType.Origin = [20, 19.5, 19.5]

      extractSurface1 = ExtractSurface(Input=clip1)
      extractSurface1.PieceInvariant = 1
      extractSurface1.NonlinearSubdivisionLevel = 1

/RunOutput/contourclip_%t.pvtp', freq=10)

    global coprocessor, pid

    if datadescription.GetForceOutput() == True:
        # We are just going to request all fields and meshes from the simulation
        # code/adaptor.
        for i in range(datadescription.GetNumberOfInputDescriptions()):
            datadescription.GetInputDescription(i).AllFieldsOn()
            datadescription.GetInputDescription(i).GenerateMeshOn()
        return

  try:
    import connectip
  except:
    print 'could not find ip address from import file'
  else:
    coprocessor.DoLiveVisualization(datadescription, connectip.connectip, 22222)
```

```xml
<sensei>
  <analysis type="catalyst" pipeline="pythonscript" filename="slice_contourcut.py" />
</sensei>
```

CATALYST LIVE THROUGH PYTHON SCRIPT





Time: 0

Statistics

Type: NA

Number of Cells: NA

Number of Points: NA

Memory: NA

Data Arrays

Name	Data Type	Data Ra

Bounds

X range: NA

Y range: NA

Z range: NA

SENSEI

insitu

http://www.sensei-insitu.org/

Sensei: A Lightweight In Situ Interface for Contemporary Infrastructure Tools and Architectures

Andrew Bauer, Patrick O'Leary and Utkarsh Ayachit

Argonne NATIONAL LABORATORY

QUESTIONS?

Silvio Rizzi
srizzi@anl.gov

Joe Insley
insley@anl.gov